NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

THE DESIGN OF A STAND-ALONE DIVISION TACTICS SIMULATOR UTILIZING NON-PROPRIETARY (OPEN SOURCE) MEDIA AND ITERATIVE DEVELOPMENT

by

Ryan B. Ernst

March 2006

Thesis Advisor:	Rudolph P. Darken
Second Reader:	S. Starr King

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: The Design of a Stand-Alone Division Tactics Simulator Utilizing Non-Proprietary (Open source) Media and Iterative Development.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ernst, Ryan B.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Fleet maneuvers, or division tactics (DIVTACS), are achieved by a series of precision shipboard movements directed by an Officer in Tactical Control. Much like a precision drill team, DIVTACS training requires multiple ships underway in close proximity, often a rare commodity. Costs to conduct live training range from several Thousand (per evolution) to several Million dollars (to repair ships after a collision at-sea). Computer simulation opens the door to maximizing DIVTACS training, while mitigating risk.

The Navy spends in excess of $60 Million per year on simulation-based training. Currently available simulators provide a DIVTACS capability by connecting several simulators together via a LAN. These simulators are cost prohibitive ranging from $100,000 to Millions of dollars per unit. They are manpower and maintenance intensive requiring dedicated infrastructures, drastically limiting deploy-ability and reliability.

Open source applications are gaining considerable leverage in the commercial market and offer significant cost-reductions. This thesis explored the possibilities of open source development by providing a proof of concept division tactics simulator. Additional considerations were given to the extension of the simulator for use in surface tactics in general and areas of future research.

14. SUBJECT TERMS Ship-handling, Virtual Reality, Virtual Environment, Surface Warfare, Computer Simulation, Computer Graphics, Open source, Division Tactics, DIVTACS, Fleet Maneuvers, Surface Tactics, Iterative Development			**15. NUMBER OF PAGES** 63
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

THE DESIGN OF A STAND-ALONE DIVISION TACTICS SIMULATOR UTILIZING NON-PROPRIETARY (OPEN SOURCE) MEDIA AND ITERATIVE DEVELOPMENT

Ryan B. Ernst
Lieutenant, United States Navy
B.A.S., Miami University, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2006

Author: Ryan B. Ernst

Approved by: Rudolph P. Darken
Thesis Advisor

S. Starr King
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Fleet maneuvers, or division tactics (DIVTACS), are achieved by a series of precision shipboard movements directed by an Officer in Tactical Control. Much like a precision drill team, DIVTACS training requires multiple ships underway in close proximity, often a rare commodity. Costs to conduct live training range from thousands (per evolution) to millions of dollars (to repair ships after a collision at-sea). Computer simulation opens the door to maximizing DIVTACS training, while mitigating risk.

The Navy spends in excess of $60 million per year on simulation-based training. Currently available simulators provide a DIVTACS capability by connecting several simulators together via a LAN. These simulators are cost prohibitive ranging from $100,000 to millions of dollars per unit. They are manpower and maintenance intensive requiring dedicated infrastructures, drastically limiting deploy-ability and reliability.

Open source applications are gaining considerable leverage in the commercial market and offer significant cost-reductions. This thesis explored the possibilities of open source development by providing a proof of concept division tactics simulator. Additional considerations were given to the extension of the simulator for use in surface tactics in general and areas of future research.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Admiral Jim Hogg and Captain Chuck Dixon for teaching me that success requires you to “fail and fail often.” Through Admiral Hogg’s inspiration I also learned that “the true power of information is openly passing it onto others” and that “it’s amazing what one can accomplish if they are not concerned with who gets the credit for the work.” These thoughts are the guiding principles by which this thesis was executed.

I would be remiss if I didn’t give thanks to those who stood by my side during this research. Dr. Rudy Darken’s positive attitude created the requisite ‘can-do’ work environment. Captain Jeff Kline’s efforts generated an invaluable early contribution from the Naval Warfare Development Center. Captain Starr King’s continued endorsement of this research as the Chair of Warfare Innovation. Captain Mark Strom’s dedicated friendship and technical assistance. The entire Delta3D Development Team for their devotion to the open source paradigm. Additionally, great appreciate goes to the Naval Education and Training Command for their continued support of innovative initiatives.

Finally, I must thank my beautiful fiancée, Miss Sarah Torrez, for her love and understanding through the most difficult times of this research.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Due to budgetary and operational constraints, live training opportunities for surface tactics are insufficient to meet the demands of the United States Navy Surface Fleet. The limited Fiscal budget must balance underway training time with logistics and equipment wear. This coupled with increased operational commitments reduces underway time for training.

Virtual training through simulation has been adopted by the Navy as a means by which to mitigate live training shortfalls. Unfortunately, commercially-available simulators are expensive to procure, operate and maintain. This factor limits the quantity of simulators available and limits simulator deployment to fleet concentration areas and the Surface Warfare Officer School in Newport, RI.

Open source applications are gaining considerable leverage in the commercial market and offer significant cost-reductions. In particular, open source offers a low entry fee, maximization of reuse, and the freedom to widely distribute, maintain and update software. The primary obstacle to fielding open source is in application development expertise. With institutions of higher knowledge, like the Naval Postgraduate School, the expertise is available to the Navy.

Seasoned surface warfare officers are subject matter experts in surface tactics. It is intuitive to train these experts to develop the applications that aspiring surface warfare officers will use in their plight for tactical proficiency. Additionally, training simulators should cover a range of sophistication with the lowest tier available to all users who may benefit from their use.

A good candidate for open source experimentation is in the area of ship-handling, particularly division tactics (DIVTACS). Live underway training in DIVTACS requires the availability of several ships underway, all of which fall under the constraints noted previously. Further, the potentially dangerous nature of DIVTACS restricts the opportunity to perform concurrent training, i.e., engineering or combat systems drills. Currently available simulators are able to perform DIVTACS, but must be networked

together. Networking these simulators is often avoided due to excessive demands for these limited resources. Instead they focus towards the more common ship-handling functions of pier-handling and underway replenishment.

Figure 1. SurfTacs Main Menu Screenshot.

B. OBJECTIVE

The primary objective of this thesis was to explore the possibilities of open source development by providing a proof of concept division tactics simulator. A secondary objective was the ability to extend the simulator for use in surface tactics in general and identification of areas of future research.

C. APPROACH

"Research through application" is perhaps the best way to describe the approach taken in this thesis. The skill set required to design and implement a ship-handling simulator cover a very broad landscape. The author earned a greater appreciation for the inter-relationships between component functionality in overall system performance through the investigation of numerous fields of study. Further, the in-depth review of the

open source paradigm provides invaluable insight into how the new "open" paradigm may be put to work for the Navy.

As will be discussed in a subsequent chapter, an iterative design was chosen to develop the prototype, appropriately named SurfTacs. Iterative design does not attempt to present a "bullet-proof" final product. Rather this design methodology focuses on generating outputs far quicker than traditional design methods, albeit less complete. The expectation is to achieve the final product after several design iterations. Every cycle affords an opportunity to learn from previous iteration mistakes and adapt the design accordingly. Major software design changes are never an easy proposition once coding has begun. However, the impact of major changes is lessened when completed earlier in the overall development process. Also, some changes may never be found until the product is released into the user population. Add in the realization that requirements often change during long development time forcing even well-designed applications to conduct a major redesign and iterative design is clearly superior to other design methodology.

SurfTacs is ultimately meant to cover an expansive set of surface tactics; from ship-handling to combat operations and from single user to multi-team training. The range of potential functionality is limited only by the objectives of the current iteration. To employ a single application in this manner requires a heavy consideration to extensibility. SurfTacs also represents a vessel through which future thesis students may choose to focus their research. To identify the limitations of the current design, offer suggestions for improvements and elaborate on opportunities for additional research, applicable chapters have been expanded with a section entitled "Future Research and Applicability to Surface Tactics (in General)."

D. SUMMARY OF CHAPTERS

The remainder of this thesis is broken down into the following chapters:

- Chapter II provides necessary background information for the current iteration of SurfTacs, in particular ship-handling during DIVTACS, the need for simulation-based training and the growing sophistication of the open source paradigm.
- Chapter III provides the overall design methodology including a more detailed summary of iterative design.
- Chapter IV discusses the virtual environment in terms of scene objects and visualizations.
- Chapter V focuses on modeling the physical environment.
- Chapter VI elaborates on the scenarios chosen for inclusion in the current iteration of SurfTacs.
- Chapter VII reports the approach taken to model human participants through artificially-intelligent agents.
- Chapter VIII identifies the intricacies of the user interface, both graphical and auditory devices used.
- Chapter IX summarizes the work of this thesis.

II. BACKGROUND

A. TRAINING SHIP-HANDLERS

Collisions between ships at sea costs millions of dollars, reduces operational availability and often ends careers of those identified as negligent in their duties. At the time of writing this thesis, the most recent ship collision occurred between two U.S. Navy Arleigh-Burke Class Destroyers, the USS MCFAUL and the USS WINSTON S. CHURCHILL. While engaged in a fleet exercise on 22 August 2005, the two destroyers collided causing over $1.3 million dollars in damage. The Commanding Officers of both ships received only administrative actions due to mitigating circumstances of the incident.[1] Even though the McFaul-Churchill collision was outside usual ship-handling operations, it is a reminder of the necessity to properly train ship-handling to prevent future collisions in all surface warfare operations.

Up until the early 1990's, all surface warfare officers received baseline ship-handling training aboard Yard Patrol (YP) craft at the Surface Warfare Officer School (SWOS) in Newport, RI.[2] However, these assets exceeded service life and funding was unavailable to replace them.[3] SWOS replaced the YPs with Bridge and CIC Team Trainers operating in a virtual environment. These simulators proved expensive to maintain and were unable to properly simulate environmental conditions and twin-screw operations needed for pier-work.[4] SWOS then procured the Conning Officer Virtual Environment (COVE) [5] as its mainstay simulator.

However, under the Division Officer Sequencing Plan (DOSP), new surface warfare officers receive the bulk of their ship-handling training on-the-job and never train with the COVE simulators until after they have successfully completed Officer of the Deck (OOD) qualifications. Prior to attending the three week SWOS course, these new surface warfare officers may benefit from the simulators at the Marine Safety International (MSI) training complexes located at Norfolk, Newport and San Diego.[6] It must be duly noted that attendance at MSI is a ship training requirement and not personnel directed thus not all surface warfare officers receive MSI training.

B. CURRENT SIMULATORS

The Cadillac of ship-handling simulators available to the Navy is the MSI training complex. MSI training focuses both on formal classroom training to teach fundamentals and the virtual environment of the simulator towards the application of the classroom training in various ship-handling scenarios. Retired Navy Captains guide teams of trainees (preferably ship designated watch teams) through both forms of training. The MSI simulation complex has both full mission bridge and bridge wing simulators. Ships are required to send at least one team to a three day and a two day training session during the inter-deployment training cycle.

The COVE ship-handling simulator provides focused conning officer training in various ship-handling scenarios. As with MSI, they do require a qualified operator to assist the training of the trainee. Unlike MSI, COVE trainers are portable and deployable. The virtual environment in COVE is projected via multiple computer screens and a head-mounted display (HMD).

Both MSI and COVE simulators may be networked to similar trainers to provide DIVTACS training. However, due to the limited availability and high demand of these simulators DIVTACS training is seldom performed with them. Instead, trainees are expected to learn DIVTACS via 2-Dimensional Maneuvering Boards (MoBoards) and on-the-job training. Being high stress multi-ship maneuvers, on-the-job training of ship-handlers during DIVTACS elevates the operational risk of these exercises. Increased risk forces additional watch-stander augmentation thereby increasing confusion and stress on the bridge.

A better method for training DIVTACS is required prior to exercising a conning officer in live training. This dilemma not only pertains to ship-handling but extends to all areas of surface tactical training. The current business models the Navy uses are not scalable to meet the Navy's demand for enhanced training. The Navy can ill-afford to produce, deploy and maintain training simulations in the quantity required to properly train junior surface warfare officers under this commercially-oriented paradigm. Something new is needed.

C. OPEN SOURCE PARADIGM

Open source is an opportunity for the creation and extension of meaningful applications that benefit the greater good. Tim O'Reilly, founder and CEO of O'Reilly Media, explains, "Early on, when software was developed by computer scientists, just people working with computers, people passed around software because that was how you got computers to do things."[7] Open source represents freedom in the information age through the concepts of unlimited distribution and open access to the underlying source code. Karl Fogel, from software distributor CollabNet said, "Freedom is a business asset, under certain circumstances."[8]

> The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.[9]

Open source developed applications are beginning to make an impact into the commercial software markets and similarly applications developed in this manner have the potential to positively impact the military. "Open source software is an idea whose time has finally come. For twenty years it has been building momentum in the technical cultures that built the Internet and the World Wide Web. Now it's breaking out into the commercial world, and that's changing all the rules."[10] "Open Source software is also gaining increased momentum in the enterprise. Commonly cited reasons for the growing interest, acceptance, and even preference for Open Source products include low cost, high value, quality and reliability, security, increased freedom and flexibility (both hardware and software,) and adherence to open standards."[11]

Further open source is about collaboration between interested parties. "The essence of the Open Source development model is the rapid creation of solutions within an open, collaborative environment. Collaboration within the Open Source community (developers and end users) promotes a higher standard of quality, and helps to ensure the long-term viability of both data and applications."[12] "No one company or individual "owns" Linux, which was developed, and is still being improved, by thousands of corporate-supported and volunteer programmers all over the world. Not even Linus

Torvalds, who started the Linux ball rolling in 1991, "owns" Linux."[13] Open source gives life to new software applications through the unity of open enterprise and is directly applicable to the desires of the military.

D. GAMING AND TRAINING

In the past computer gaming received the oppressive stigma of existing just for "mindless entertainment." Today, computer gaming is a multi-billion dollar enterprise and is firmly entrenched within the culture of our youth. One of the initial steps in creating software of any kind is to understand the intended users. 60% of Americans interact with computer games with an average age of 28. 43% of gamers are female.[14] Since the military is a reflection of society, it stands very likely that a large portion of new recruits participate in computer gaming.

Computer games have evolved from simplistic amusement. They now represent an opportunity to conduct immersive training in a virtual environment that intrinsically appeals to the curiosity of the trainee.[15] By leveraging a trainee's curiosity, the trainee becomes personally motivated to learn. This motivation creates an attitude of intentional learning resulting in the retaining of useful information.

> An attitude of intentional learning — of investing extra mental effort, beyond what is required just to complete a task, with the intention of achieving personal goals for learning — is a problem solving approach to self-education because the goal is to transform a current state of personal knowledge (including ideas and skills) into an improved future state. Effective intentional learning combines an introspective access to the current state of one's own knowledge, the foresight to envision a potentially useful state of improved knowledge that does not exist now, a decision that this goal-state is desirable and is worth pursuing, a plan for transforming the current state into the desired goal-state, and a motivated willingness to invest the time and effort required to reach this goal.[16]

Military training does not have to be a laborious and non-enjoyable experience. By utilizing game-based training, military education may benefit from the trainee's curiosity and self-motivation to learn. The inspiration to train via gaming may even apply outside of normal working hours if the virtual environment is appealing to the trainee. Game-based training is an area that should continue to be explored and exploited by the military to address the educational needs of today's warriors.

III. OVERALL DESIGN

A. DESIGN METHODOLOGY

Design methodology is essentially the philosophy by which software is developed. There are several competing design methodologies currently in practice. They range from the traditional waterfall method to the iterative design process. The various design methodologies will be briefly discussed in the following passages.

1. Waterfall Design Model

The waterfall model is a sequential process which seeks to forward a finished product through each step in the software lifecycle. If problems are identified in the product by the subsequent step, the process takes a step backward and reworks the product. Thus, waterfall designs tend to be linear in execution and often yield very long software development times. Additionally, the waterfall model is very inflexible to late changes in software requirements. It is not unheard of for software developed in this manner to be out-dated upon completion.[17]

2. Spiral Design Model

Spiral design seeks to mitigate some of the drawbacks of waterfall design by executing the sequential steps several times. Spiral design also places emphasis on assessing and mitigating risk between execution cycles. Changes in requirements tend to not be as detrimental in this process. This spiraling pattern takes place primarily within the software developer. Like the waterfall method, the product released to the customer is expected to be complete and takes a considerable amount of time to produce.[18]

3. Iterative Design Model

Iterative design seeks to mitigate the drawbacks of spiral design by providing intermediate products to the consumer. Many errors in software will never be found by the engineers who create them. These deficiencies are not necessarily programming errors, but rather are failures to properly meet the needs of the customer. Non-intuitive interface design is a primary example of this failure. Since both waterfall and spiral design methods never release the product until it meets the requirements and specifications documents, many of these errors are never found. In the case of the military, software produced in this manner may be shelved or a follow-on design ordered

to correct these discrepancies. Since iterative design passes a product to the user sooner, albeit less-than-complete, these failure are found earlier in the development process and may be corrected during subsequent iterations.[19]

B. EXTENSIBILITY

SurfTacs is a very ambitious project that will never be completed. That is to say, as long as there is interest in the continued development of SurfTacs, the design will iterate indefinitely, referred to as the "continuous beta." A pitfall in iterative software development is to "paint yourself into a corner." In other words, work in a current or past iteration prevents the success of future iterations. To avoid this tripwire, the software development must seek to apply the software concept of extensibility. By definition, extensibility is the capability of being extended.[20]

Object-oriented Programming (OOP) provides the opportunity for extensibility through the use of abstraction, encapsulation, inheritance and polymorphism. Data abstraction provides a mechanism to focus on the interface between objects and the ability to ignore the details within an object. Encapsulation provides a means to hide information from external objects. Inheritance permits specialization and, through careful design, the opportunity for genericity. Polymorphism permits a family of classes to utilize a singular interface thereby providing a means to execute unique behavior by subclasses without the requirement for explicit knowledge of the subclass instance.[21]

C. CURRENT ITERATION

SurfTacs follows the iterative design methodology. It is assumed this program will not meet all requirements of surface tactics in its initial iteration. The decision to focus on DIVTACs was made to provide a product to the Navy serving a greatly needed void in the training of junior surface warfare officers. The author's surface ship experience was also instrumental in this decision. Future iterations of SurfTacs will expand beyond the constraints of the bridge watch into all areas of surface tactics and will also serve to correct identified deficiencies of the previous iterations. As mentioned previously, extensibility must be carefully considered to avoid constraining future iterations.

D. APPLICATION PROGRAMMING INTERFACE HEIRARCHY

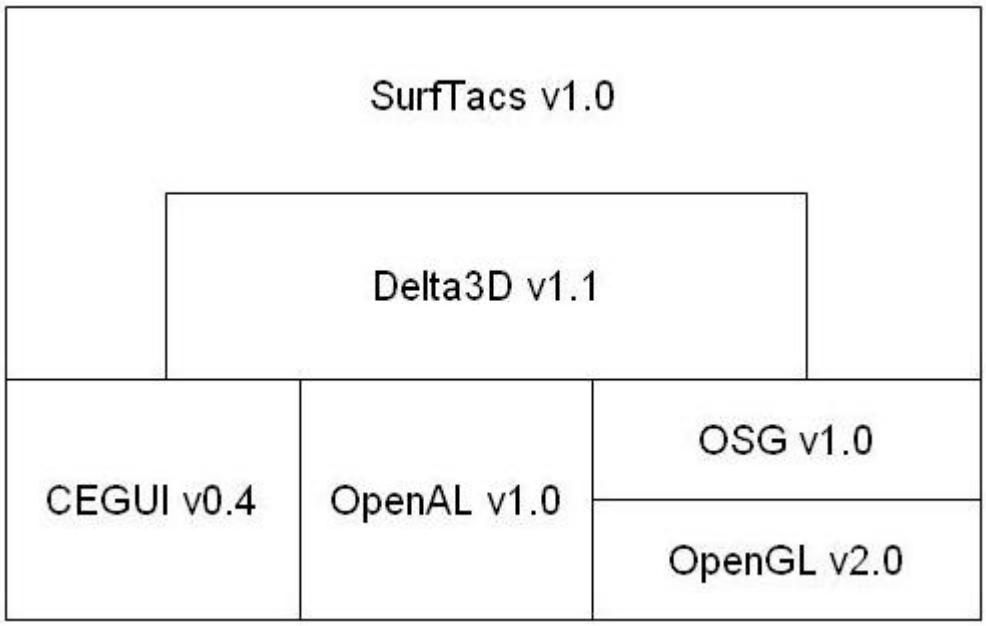

Figure 2. API Hierarchy.

Figure 1 illustrates the hierarchy of interactions between SurfTacs and various application programming interfaces (API). At the heart of this hierarchy is Delta3D v1.1. Delta3D "is a full-function game engine appropriate for a wide variety of modeling & simulation applications."[22] Delta3D is funded by the Naval Education and Training Command (NETC) and is under development at the Naval Postgraduate School in the Modeling and Simulation Institute (MoVES). Delta3D is written in Standard C++ and merges various independent open source initiatives into a higher-level API. Delta3D significantly aids open source application development by abstracting the details of lower-level API while preserving the capability to transcend directly to these lower-levels as required.

CEGUI v0.4, short for Crazy Eddie's GUI System, "is an open source library providing windowing and widgets for graphics APIs / engines where such functionality is not natively available, or severely lacking. The library is object orientated, written in Standard C++, and targeted at games developers who should be spending their time creating great games, not building GUI sub-systems."[23] CEGUI was selected over other

open source GUI APIs due to its extensive capabilities. Additionally, CEGUI is aggressively under development insuring future applicability.

OpenAL v1.0, “is a cross-platform 3D audio API appropriate for use with gaming applications and many other types of audio applications.”[24] OpenAL handles the intricate details of audio hardware manipulation and allows the application to focus instead on listener and sound positioning and play-back. OpenAL is written in Standard C++. Delta3D adds instrumental audio management functionality easing the process of application audio insertion.

Open Scene Graph (OSG) v1.0 “is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modeling. Written entirely in Standard C++ and OpenGL it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris and FreeBSD operating systems.”[25] OSG provides robust scene graph functionality required of advanced visual simulation. Additional OSG utilities include; file loading, particle system effects, and many others.

OpenGL v2.0 “is the premier environment for developing portable, interactive 2D and 3D graphics applications.”[26] OpenGL handles the intricate details of video hardware manipulation and allows the application to focus on graphical object creation, positioning and updating. OpenGL is written in Standard C++.

E. SURFTACS LAYOUT

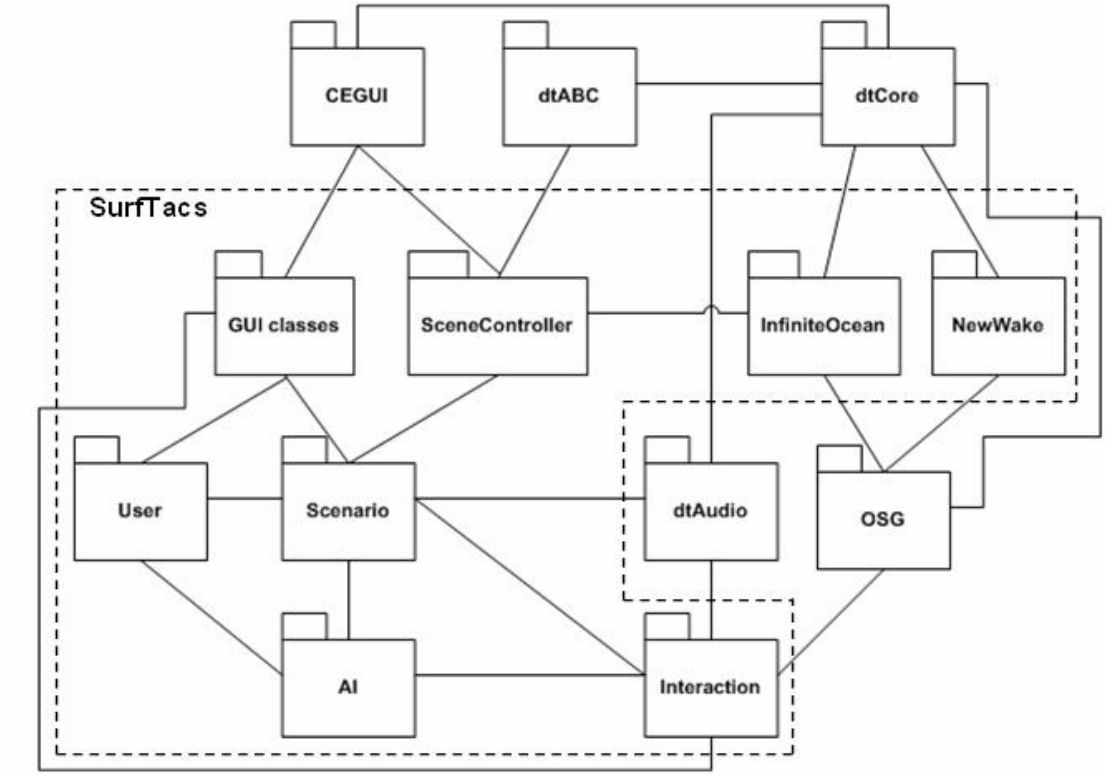

Figure 3. High-level Design.

The Unified Modeling Language (UML)[27] is leveraged to display the elements of SurfTacs and their collaborations with one another and with supporting API elements. Figure 2 illustrates the high-level design of SurfTacs. Note packages in this diagram may represent; an independent class, a major sub-component of SurfTacs, a module of an API, or an entire API. This high-level design serves as a roadmap for subsequent area-specific chapters in this thesis. In order to properly orient the reader, Figure 2 should be referred to prior to commencing each chapter.

F. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

It is often easier to visualize a better path once you've reached your destination. This is definitely the case with the overall design of SurfTacs. The author utilized pointers liberally to create collaborations between classes. This structured approach is reasonable for small applications, but quickly turns into a nightmare for moderate and

larger applications. Often called “spaghetti code”, overuse of pointers makes graceful deletion considerably more challenging and generates difficult to understand code.

A message-based design is one method that could be employed to avoid liberal pointer usage. In a message-based design, messages are sent to specific classes vice direct class method calls. If properly designed, the message sender will be unaffected if the intended recipient is unable to receive the message (i.e., the receiver has been deleted). Delta3D provides basic message functionality in all classes derived from the Base Class. Deriving from Base also yields the inherit option to reference class objects further aiding graceful deletion.

The author highly encourages a complete redesign of SurfTacs based on a message-based design. It is equally recommended to incrementally add functionality while insuring the graceful removal of the added functionality. This approach will significantly reduce painful implementation efforts to localize memory leaks and prematurely deleted references.

IV. SCENE OBJECTS AND VISUALIZATIONS

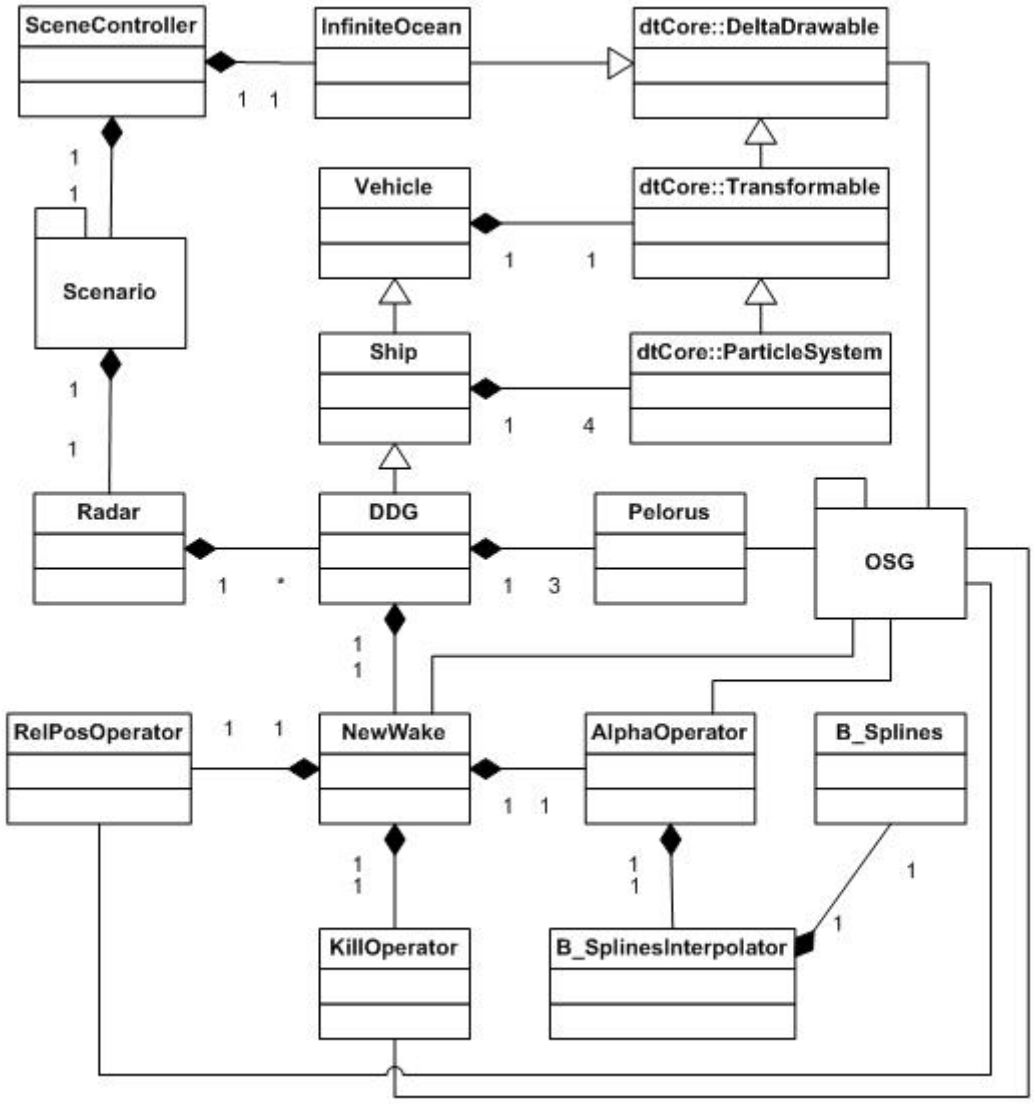

Figure 4. Scene Objects Design.

A. ENVIRONMENT

1. Weather Modeling

Delta3D offers considerable native support for creating realistic visual environments. In particular, creating an instance of dtABC::Weather provides an eye-pleasing sky dome with a variety of options for cloud cover, time-of-day and fog

intensity. Fog is especially important in SurfTacs as it naturally fades distant objects into the horizon, generating a visual distance reference for the user. In order to receive the benefits of fog, a visual object must be added as a "child" to weather. For clarity, Figure 3 omits dtABC::Weather and the "child" associations to the other classes.

2. Ocean Modeling

Unfortunately, at the time of designing and implementing SurfTacs, native Delta3D support for water (in particular an ocean) was unavailable. This area is critical to the believability of a ship-handling simulator. The question of whether to design a 3D ocean (render realistic waves) or a 2D abstraction quickly came to the forefront. 3D waves are nothing new to ship-handling simulators. MSI has had this functionality for a number of years. However, this feature is rarely used in practice due to the likelihood of inducing "simulator sickness" in trainees. Additionally, 3D ocean models require considerable CPU processing time to work correctly.

In order to avoid creating a costly feature (in terms of development time and CPU performance) that would likely be avoided by the end-users, the decision was made to create a representative 2D ocean. The end result was a large textured plane that "moved" with the eye-point. The texture was repeated numerous times and "pulled" across the plane to generate the effect of wave movement and eye-point velocity. Combined with some OpenGL blending, the ocean appears believable and is in keeping with the affinity of the remaining visual elements of the application.

B. SHIP MODELING

1. File Formats

OSG offers support for numerous file formats. SurfTacs utilizes OSG (ASCII text) and IVE (binary) formats. The OSG file format, being a human-readable format, was particularly useful in making direct changes to the file without the assistance of a modeling program. OSG files were utilized for the ship stack heat and bow and rooster wakes (more on these visualizations later in this chapter).

The IVE file format is a native binary format added to OSG as a plug-in. The IVE binary plug-in, developed by Uni-C's VR-Center and submitted as open source, adds support for binary reading and writing OSG nodes. IVE format produces a much faster

load time (10-20 times) and smaller file size then the native ASCII OSG format. Since IVE is a runtime format, it is important to keep original files, OSG, FLT, 3DS, etc., in order to modify the models in the future.[28]

Figure 5. SurfTacs Bridge Screenshot.

2. Guided Missile Destroyer

The centerpiece of SurfTacs is the Arleigh-Burke Class Guided-Missile Destroyer (DDG) model. The original model was created by a student in the NPS MoVES curriculum (unknown name). Several modifications to the model were required in order to create the functionality required for SurfTacs. In particular, a bridge area was added to the destroyer model. Several visual objects on the bridge were also added (radar, helm console, CO/XO chairs, etc). The DDG model further represents the power of open source as the improvements made to the original model are now available for modification by others.

Pelorus and rudder indicators were later added independent of the destroyer model in order to facilitate dynamic manipulation. It should be noted that this functionality may be added directly inside the model hierarchy and later manipulated

inside the application code (via various OSG nodes). However the author's limited modeling experience drove the decision to instead implement these items directly inside the application code.

C. PARTICLE SYSTEMS

1. Ship Stack Heat

Delta3D offers native support for the creation of particle systems (psEditor.exe) and the insertion and manipulation of particle systems within the engine. The gas turbine engines (which both create the needed force to turn the propellers and produce electricity in the Arleigh-Burke Class DDG) generate substantial amounts of heat. This heat combined with small amounts of smoke creates a subtle visual effect in the vicinity of the stacks. A particle system was designed to recreate this effect and loaded into the application using the OSG file format (discussed previously in this chapter).

2. Ship Wake

The ship's wake proved to be a little more challenging to implement. Initially, the wake was created using the resources available natively within Delta3D. Separate particle systems were used to create bow, rooster tail and stern wakes. The available options within Delta3D offered a reasonably realistic wake effect. Everything was fine until the author encountered floating point errors as a result of moving the eye-point too far from the origin. The floating point error problem occurs due to the limitations of numerical precision. Not all floating point numbers are possible and due to the promulgation of error (through repeated floating point math) as fewer bits are available to the right of the radix (as in moving away from the origin). The visual effect which occurs is jitter, irregular appearing movement, which increases in fluctuation quickly to the point of unacceptability.

There are numerous ways to correct the floating point error problem in virtual environments. One way is to utilize higher precision numbers (for instance type double) for all vertex positions. Naturally, this is only a patch to the problem as eventually the same problem will occur, though in the case of the needs of SurfTacs this may not have been an issue. The second method is to maintain the eye-point at the origin and instead move the world about the eye. This change would have caused a major design alteration and, since the problem wasn't identified until well after implementation had begun, was

not considered a desirable solution. A third way to correct floating point error was to "reposition" all entities when the eye-point passed outside some arbitrary distance from the origin. Through empirical research, 1500 meters was chosen as this distance.

Utilizing the reposition method solved the jitter problem. Unfortunately it also created a new problem: isolated particles. Delta3D offered no direct support to manipulate particles directly. The solution ultimately was to create the particle system directly in OSG and derive additional classes from osg::Operator and osg::Interpolator. The derived operators provided a mechanism to manipulate individual particles, i.e., position, velocity, life (energy), etc.

Additionally, the author (assisted by Captain Jeff Wrobel, USMC) implemented a B-Spline interpolator for the alpha blending of textures in order to maximize the visual effect of wake particles while maintaining a natural-looking fade out. These measures were applied to the stern and side wakes, leaving the bow wake and rooster tail untouched as they are both short duration effects and observation of their isolated particles after a reposition is trivial.

D. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

OpenGL Shaders offer an excellent opportunity to create a more realistic 2D ocean than the one designed and implemented into SurfTacs. Shader support has recently been added into Delta3D and some initial work towards using OpenGL Shaders to create a realistic 2D ocean was done by the development team. It remains to be seen if native support for procedurally-created oceans will be added to Delta3D. As this is an area that will likely resurface during a design iteration of SurfTacs, the performance-minded modeling of oceans is an excellent area for focused student research.

Creating better models for SurfTacs is also an area of needed work. In particular, one may speculate the future desire to transverse through the internal passageways of the destroyer. Why limit to SurfTacs to Arleigh-Burke class destroyers? Additional ship models may also be desirable for future iterations of SurfTacs. In visual modeling, the canvas has an infinite number of possibilities and is limited only by the imagination of the modeler and the affinity of the model to the remaining visual objects of the simulation.

V. PHYSICAL WORLD MODELING

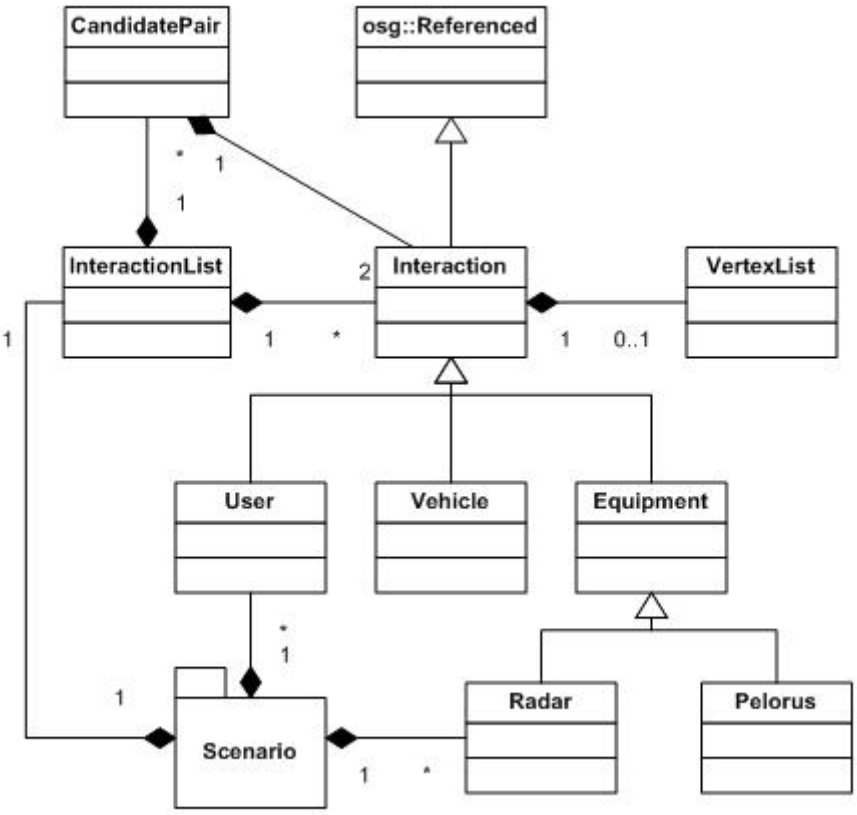

Figure 6. Physical World Design.

A. SHIP MOTION

Physical world modeling deals with how virtual objects interact with the virtual environment. For this iteration of SurfTacs to appear realistic to the user, the ship model must behave in a reasonable manner with dynamic changes in rudder and throttle. Here is yet another area where an entire thesis research may be based. However, in keeping with the "mile wide and an inch deep" philosophy of this thesis, the author chose to utilize a fairly simplistic motion model based on a few key assumptions:

First, SurfTacs is meant to be openly available to all that desire its use. Thus, only unclassified information may be leveraged to drive the physical model. Without precise (and classified) ship characteristic data, an upper bound is quickly placed on the

realism of the physical model. However, utilizing as a base for calculation the unclassified knowledge that a ship's tactical diameter approximates 1000 yards with all-engines ahead standard and standard rudder (for smaller ships), combinations of engine order and rudder may be applied in a representative manner. This works out to be a reasonable approximation for engine order and rudder combinations close to standard, but will likely lose accuracy as the engine order and rudder combination varies from this base.

Second, the scenarios (to be discussed in a following chapter) exercise DIVTACS. The proper execution of DIVTACS by individual ships most often requires the use of standard rudder and speeds greater than bare-steerage and up to stationing speed. If a base speed of fifteen knots (standard engine order on a destroyer) is ordered for the formation, on average the physical model will be a reasonable approximation

Third, relative motion between ships is what is most important in DIVTACS vice single ship motion with respect to the environment. Since all ships will essentially be affected similarly by environmental effects like current and wind, these forces may be abstracted with minimal loss to the fidelity of the physical model. This would certainly not be the case in scenarios where environmental effects or the interaction between ship forces apply, i.e., pier-handling, underway replenishment, etc.

B. USER MOTION

In this iteration of SurfTacs, user motion is independent of ship motion. User maximum speed is modeled as 2 m/s, the equivalent of a brisk walk. The user may freely move throughout the bridge area of their destroyer. The user is, however, confined to their ship. Abstracting the motion of the ship (permitted through transform matrices and use of 'child' relationship in Delta3D), user motion is essentially a 2D problem. With this in mind, the author pursued the collision detection and handling scheme that follows.

C. COLLISION DETECTION/HANDLING

Early in the application development, a decision was made to create a specialized collision detection scheme for SurfTacs. At that time, Delta3D was also in an early state of development and had recently added the Open-Dynamics Engine (ODE).[29] The stability of ODE in Delta3D appeared somewhat questionable. In retrospect, the Delta3D team overcame the integration challenges and now ODE is an excellent contribution to

the Delta3D simulation engine. Like many naive software designers, the author chose to reinvent the wheel by creating a collision detection scheme from scratch. This decision was perhaps the worst design decision made by the author. The following describes this collision detection scheme to aid future SurfTacs developers/maintainers.

The Interaction class is the base class for collision detection. An instance of Interaction may be of type dynamic, static or proximity. Dynamic Interactions have a corresponding dynamic transform capability and thus may collide into other Interactions. A Static Interaction has a static transform and does not actively collide into any Interaction (even if positioned within the boundaries of another Interaction). Both Dynamic and Static Interactions represent a physical object. A Proximity Interaction is similar to a Static Interaction, but does not represent a physical object and is used to trigger an event, i.e., user interaction with a piece of equipment (more to follow on interfacing with equipment in the chapter on User Interface).

An Interaction List is a container class for all Interactions within a similar virtual space. This permits the ability to have several Interaction Lists to segregate Interactions. For example, this iteration of SurfTacs has two Interaction Lists; the recognized maritime picture (RMP) for ship-ship collisions and the bridge area to handle user collisions with the bulkheads, equipment and proximity sensors. The Interactions contained within separate Interaction Lists do not influence one another.

Inside an Interaction List, potential collision between Dynamic and other Interactions are identified and stored as Candidate Pairs. Utilizing a bounding circle (the collision detection scheme is 2D) and a maximum speed (for Dynamic Interactions only), the minimum possible time for a collision to occur between the Interactions within a Candidate Pair is derived from their actual distance apart. The minimum possible time is added to the current time and assigned to the Candidate Pair update time. When the current time exceeds the Candidate Pair update time, the candidate pair is checked for collision. If a collision has yet to occur, a new update time is calculated as performed above.

In the event a collision occurs, the Dynamic Interaction is alerted to the collision with a pointer to the Interaction the object collided with. In the case of a ship-ship

collision, the training objective of the scenario is unable to be met and a message to the user is presented alerting to the failure. However, when a user collides with an object on the bridge, the show must go on. A very simple collision handling scheme was used to handle collisions of this manner. The user's motion (at the time of collision) was simply reversed and multiplied to ensure the user Dynamic Interaction cleared the boundary of the object with which it collided.

The user collision handling scheme leaves much to be desired. First, unless the user collides with the object at a perpendicular angle, the reversal in direction is precisely opposite of the incident angle to the plane of the object. This is not only counter to what the user expects but has the added discrepancy of prohibiting the user from smoothly traversing along a plane. The second deficiency of this collision handling scheme is the inherent possibility of getting "caught" inside the boundary of another Interaction when blindly applying motion reversal. The result freezes the user in place which is certainly an undesirable to place for the user to be! Getting "caught" seldom occurs in practice, but even a single occurrence destroys user confidence in the believability of the application.

D. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

After the discussion on collision detection and handling, it should be quite apparent the need for a major design change to SurfTacs is required. Additionally, 3D collision detection will be necessary for any future iteration of SurfTacs where movement is outside the 2D plane, i.e., a scenario where shells are fired at an approaching aircraft. A more generalized and efficient approach to collision detection is to implement ODE. Collision handling may be performed through ODE or properly performed within the application-specific code.

The ship model used in the current iteration of SurfTacs is acceptable for DIVTACS. However, the extension of SurfTacs to other areas of ship-handling requires a considerable improvement to both the physical model of the ship and the virtual environment. Environmental forces acting on the ship, current and wind for example, are important when an object unaffected (or affected differently) by these forces is added to the scene, i.e., a pier or a man in the water. Further, the ship itself imparts a force on the

surrounding environment. This force manifests as high pressure zones at the bow and stern and low pressure zones along the sides of the ship. These forces are particularly important to the interaction between ships at close proximity (through the application of Bernoulli's principle.)[30]

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SCENARIOS

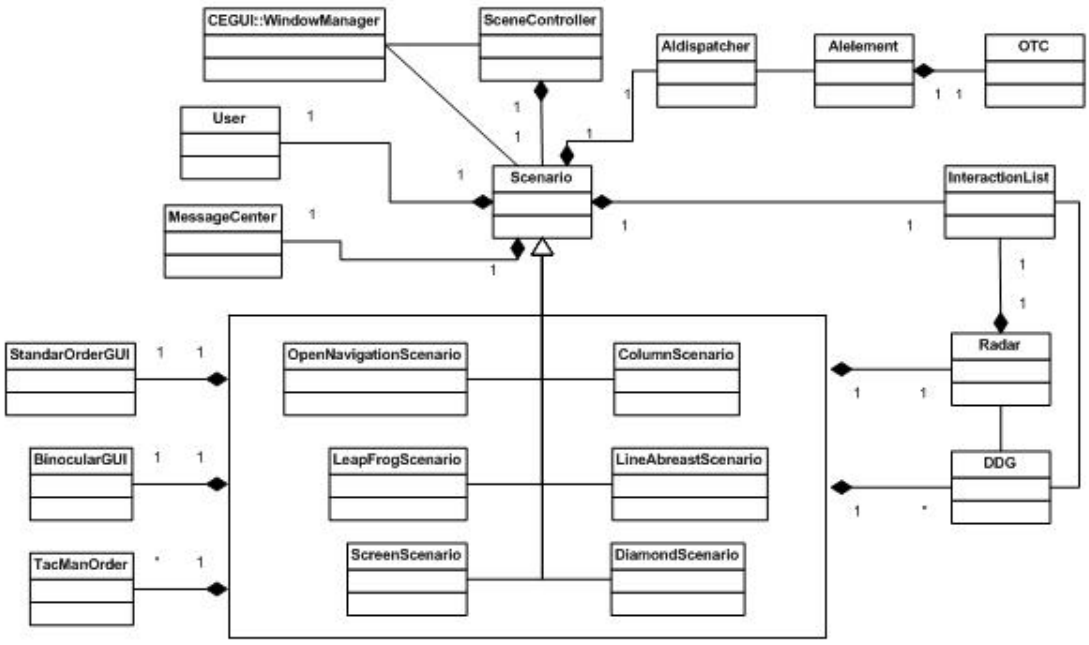

Figure 7. Scenario Design.

A. INCLUDED SCENARIOS

In SurfTacs, specific scenarios are derived from the abstract Scenario Class. Aside from general application setup, persistent GUI elements and the environment, each scenario is independent of any other scenario. Designing the scenarios in this manner maintains the desirable attribute of extensibility. The first scenario selected takes the longest to load as the environment must be created and added to the scene. However, within the same session of SurfTacs, all other scenarios load extremely quickly. In order to avoid classification issues, maneuvering signals are in plain text and are of a representative nature only. The scenarios included in this iteration of SurfTacs represent a portion of DIVTACS but do not cover the entire breadth of DIVTACS. The following is a breakdown of the scenarios included:

1. Open Navigation

The open navigation scenario permits the user to explore the bridge of their destroyer without the distraction of additional shipping. This scenario is open-ended and

is thus completed when the user elects to finish. This scenario is meant for a first-time user and is not expected to be utilized more than once.

2. Leapfrog

The leapfrog scenario is based on a Navy training exercise to develop ship-handling skills for underway replenishment. In this scenario the user's destroyer is placed astern a second destroyer. The order to "take station" is given by the Officer in Tactical Control (OTC). The user is expected to place their destroyer alongside the second destroyer at the prescribed distance listed in the scenario text. The distance is sufficiently large to avoid the Bernoulli effects created between ships at close proximity. The second destroyer will not alter course or speed for the entire scenario. The scenario is completed when the user's ship maintains position alongside (within the tolerances of +/- 3 degrees and +/- 10% range) for an uninterrupted 60 seconds. The leapfrog scenario is perhaps the best scenario for training the effects of relative motion.

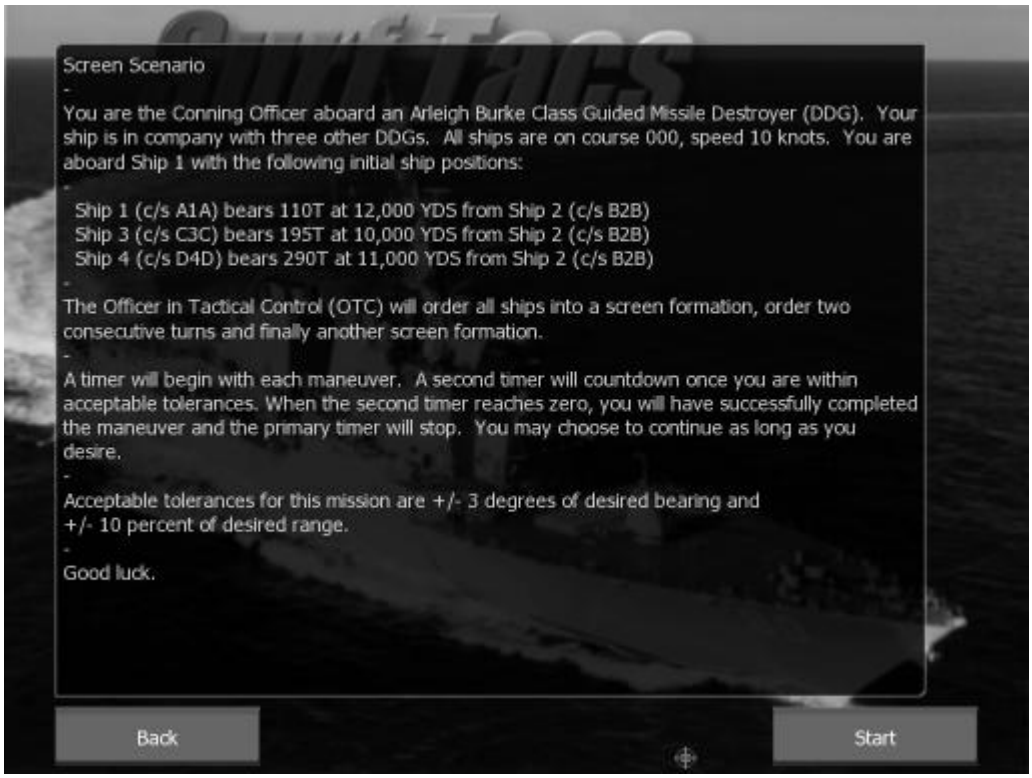

Figure 8. SurfTacs Scenario Screenshot.

3. Screen Formation

The screen formation scenario creates fours ships and places them apart at a considerable range from one another. The OTC sends a signal to form a screen formation and all ships are to head to their assigned screen sectors. Three of the four ships are conned by artificially intelligent agents. The guide ship maintains course and speed while the other two destroyers alter speed and course to properly execute the tactical signal. In a manner similar to the Leapfrog scenario, the signal is completed when the user's ship achieves and maintains station for an uninterrupted 60 seconds. Three other tactical signals are ordered in succession; two standard turns and finally an additional screen formation. At the successful completion of the final tactical signal the scenario is completed.

4. Column Formation

The column formation scenario is a continuation from the screen formation scenario in that all four ships are initially placed in a screen. The OTC then orders all ships into a column formation. Upon completion of the signal, the OTC follows with an order to conduct a wheel turn and finally a standard turn is ordered.

5. Line Abreast Formation

The line abreast formation is a continuation from the column formation scenario in that all four ships are initially placed in a column. The OTC then orders all ships into a line abreast formation. Upon completion of the signal, the OTC follows with an order to conduct a wheel turn and finally a standard turn is ordered.

6. Diamond Formation

The Diamond Formation is also a continuation of the column scenario in that all four ships are initially placed in positions loosely resembling a column. The first order given by the OTC is to formally place all ships into a column formation. From the column, the OTC orders the diamond formation. From here two standard turns are ordered and the scenario is completed.

B. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

Future scenario work may certainly include additional DIVTAC maneuvers. However, scenario generation may include areas outside this realm of training. For instance, with minor modifications a scenario could be created to exercise open ocean navigation and application of rules of the road procedures. Perhaps man overboard, torpedo evasion, small boat operations or flight operations could be included. Further, tactical warfare scenarios like small boat attacks may be implemented. The possibilities are limited only by the imagination and time available to the application designer.

Deriving scenario classes is the current method for handling scenarios. However, this method restricts the addition and modification of scenarios to those compiled within the executable. An alternative approach is to utilize external files (i.e., XML files) for this purpose. XML adds the advantage of an easy to understand file format which would greatly increase the ability of the end-user to extend the scenarios in SurfTacs without the burden of compiling a new executable.

VII. ARTIFICIALLY INTELLIGENT AGENTS

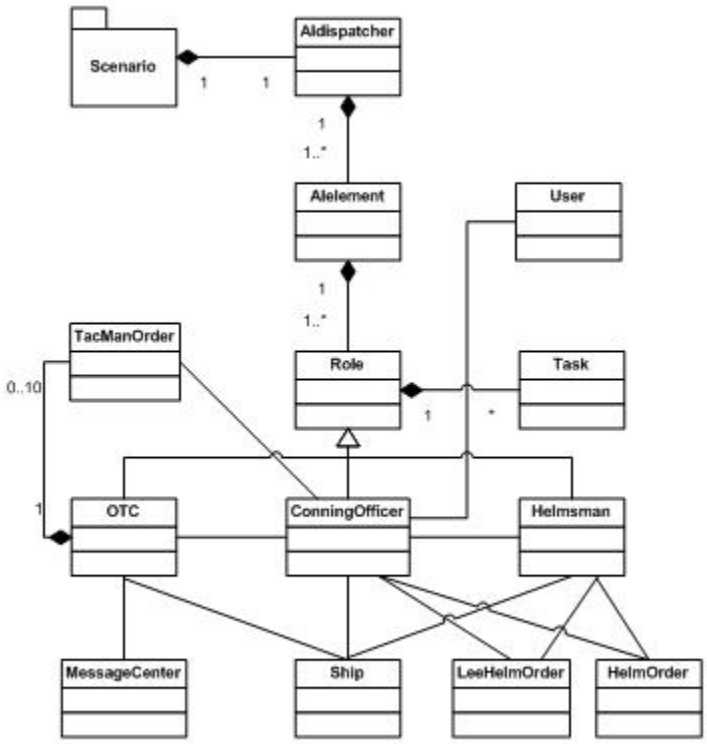

Figure 9. Artificial Intelligence Design.

A. DISPATCHER

Taking a top-down look at the Artificial Intelligence (AI) design of SurfTacs, the Dispatcher class is resident at the highest level. The Dispatcher class in the current iteration of SurfTacs is nothing more than a container class of pointers to AIelement classes. An instance of Dispatcher authorizes assigned AIelements to update each cycle. The original intent for this class was to add the ability to prioritize the AIelements in order to distribute scarce processing resources. However, in this iteration AI processing resources did not significantly impact frame times to justify the additional design work to implement this feature.

B. AI ELEMENT

AIelement derived classes are the primary agents within SurfTacs. The heart of this class is a time-based priority queue of Tasks (discussed later in this chapter). To make the AIelements appear more realistic a maximum of one task is permitted every cycle. Additionally, tasks may be added with a time delay. When the current time is greater than the first task in the queue, the associated AIelement and, more specifically their assigned Role (more on this class in the following section), is notified of the pending task and the Task is removed from the queue. Tasks may also be prematurely removed from the queue by direction of the AIelement or by the associated Role. This capability is particularly useful if a new task is generated that conflicts with a previously added task.

C. ROLE

A Role in SurfTacs is a watch stander. For example, the ConningOfficer class is a Role that is assigned to AIelement. Additional derived classes from Role include Helmsman and Officer in Tactical Control (OTC). As in actual Navy command structures, SurfTacs employs a watch standing hierarchy through the use of orders. There are three types of orders embedded in this iteration; HelmOrder, LeeHelmOrder and TacManOrder. The first two orders are understood between each set of conning officers and helmsmen while the third order is used between the OTC and all conning officers.

D. TASK

As mentioned previously, Tasks are created by Roles and inserted into the time-based priority queue located in the associated AIelement. Tasks are essentially sub-routines performed to accomplish a desired end state. For instance, one Task in the Helmsman Role is to “maintain course.” The goal of using Tasks is to reduce a routine into as many sub-routines as possible. This aids both reuse of Tasks in various combinations and the realism of the AI in the simulation due to the limiting aspects of performing only one task per cycle (see AIelement above).

E. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

A natural extension to the AI in SurfTacs is the addition of more Roles. In keeping with past recommendations, future iterations of SurfTacs may venture outside of the scope of bridge watch-standing. For example, the creation of a Role to simulate a .50 caliber machine gun operator would be very useful in a scenario depicting a small boat attack. As with other areas of SurfTacs, the possibilities for additional Roles are seemingly endless.

Dedicated future AI research could better associate the SurfTacs AIelement with the Human mental model. Creating more realistic AI starts with a better understanding of the limitations of Human Beings. By leveraging ongoing research in Human Factors Engineering, AI agents could be created with similar limitations as the Humans they are attempting to depict. Areas to consider in the AI mental model could include the time to complete a task, the ability to hold items in both short term and long term memory, and also the application of the concept of attention.[31]

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. USER INTERFACE

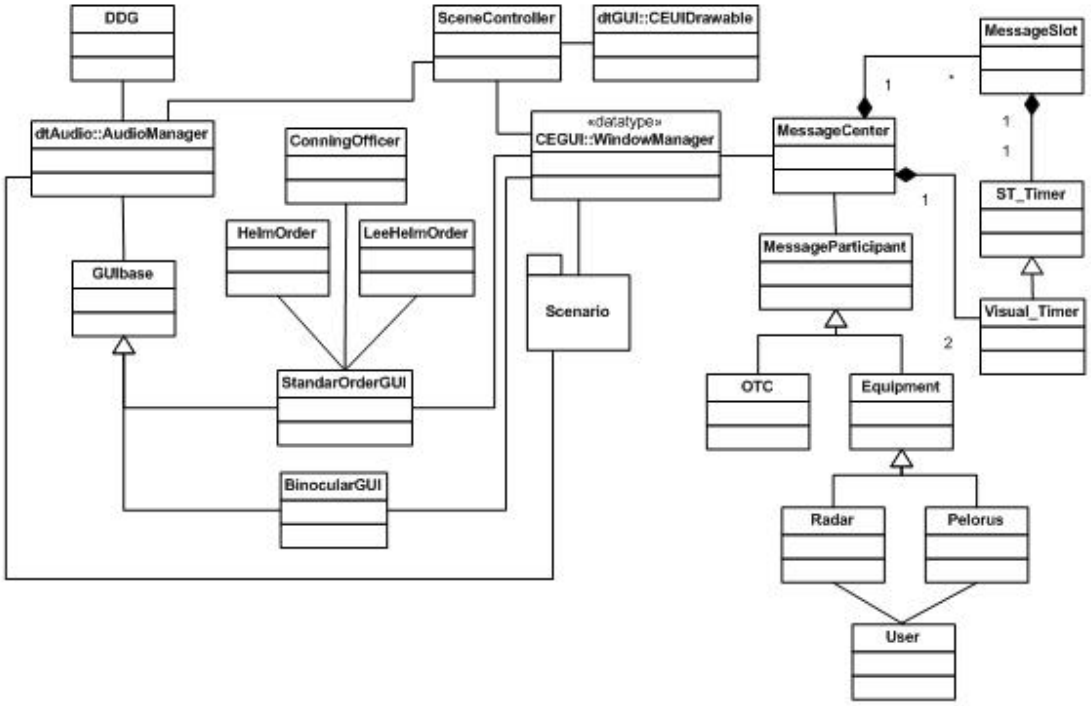

Figure 10. User Interface Design.

A. REAL WORLD VS VIRTUAL ENVIRONMENT

Recall the military adage, "Everything short of war is simulation." This bold statement reminds application designers that no matter how hard they try to build realism into simulations, they will be less than real. Applying this knowledge to application design is to repeatedly ask the question, "What is the training objective?" Just because the technology available permits something to be done, does not mean it should be done.[32] An example is the tradeoff between 2D and 3D ocean models (discussed in chapter 14). Areas of interest with respect to user interface include; dialog between the user and AI agents and user interaction with equipment.

In this iteration of SurfTacs, the user must communicate with three AI agents; the OTC, the helmsman and lee-helmsman. In the actual ship-handling environment, both utilize verbal communication. However, dialog between the OTC and the conning officer is through tactical radio communications. The skills associated with radio

communications are both physical and cognitive. As mentioned in chapter 6, the tactical orders provided by the OTC are in plain language to avoid classification issues. This communication is further abstracted to the simulation user in the form of a dialog box. The user's response to a tactical order is restricted to only "Roger, out" meaning I understand the order and will comply. The interaction focuses on the training objective of executing a received tactical order while abstracting the cognitive ability to break a coded order and generation of a proper reply. Additionally, the physical ability to utilize verbal communications with the tactical radio is similarly abstracted.

Dialog between the user and the helmsman/lee-helmsman is considerably more challenging due to the complexity of a standard order to the helm and the necessity for two-way dialog. A standard naval order may consist of a helm order and/or a lee-helm order. Internally, SurfTacs divides these into separate orders for simplicity (a HelmOrder to the Helmsman and a LeeHelmOrder to the LeeHelmsman). However, this abstraction is not directly apparent to the user. To generate a standard order the user uses a graphical user interface (GUI) consisting of buttons, sliders and text boxes. The user has several ways to input a desired standard order. This versatile interface provides a reasonable representation of the cognitive requirements to generate a standard order but fails to exercise the user's physical ability to properly execute the order verbally. Additionally, the dialog between the conning officer and helmsman/lee-helmsman is captured in a message window without auditory feedback. This abstraction provides a minimal visual alert to the user and may not properly alert the user to important feedback from the AI agents.[33] For a recommended solution to this discrepancy, see the final section of this chapter on future research.

Two forms of equipment are included in the current iteration; pelorus and radar repeater. All forms of equipment may be "engaged" when the user is within a predetermined proximal distance. There are three pelorus located on the bridge; one on each bridge wing and a third at the centerline. A pelorus provides a means to ascertain relative bearings, true bearings and relative motion between own ship and a visual reference point. The pelorus must be actively disengaged by the user, but remains in its final position giving a quick reference to the user while permitting freedom of motion. The radar repeater provides an abstraction of the radar picture in the form of a textual

listing of all ships and their bearings and ranges in the message window (discussed in the following section). This abstraction was chosen to focus on the cognitive ability of assessing the surface picture vice the physical ability of detection, classification and tracking of surface contacts.

Figure 11. SurfTacs GUI Screenshot.

B. GRAPHICAL USER INTERFACE

SurfTacs benefits greatly from a graphical user interface (GUI) made possible through the use of dtGUI and the CEGUI API. The GUI consists of a menu system intended to provide information to the user and the ability to select, start, pause, restart and quit the included scenarios. The GUI also provides multiple views to the user and annotated by symbolic icons including; default mouse mode with a mouse icon, a standard order window identified by a ship's wheel icon, and a binocular view associated with a binocular icon. These icons were chosen based upon the results of a visual design survey and are in keeping with the visual design principle of meaningfulness.[34] Finally, the GUI provides a messaging system to the user in the form of a dynamically changing text window and a pop-up style non-modal dialog box.

C. AUDITORY CUEING

Through dtAudio and the OpenAL API, several sounds have been added to SurfTacs. Upon entering the application, the user is greeted with four bells, often signifying the return of the Commanding Officer. The use of this audio clip combined with an exciting background image of an Arleigh Burke Destroyer[35] underway is meant to excite the user prior to commencing a scenario. Another short audio clip is used whenever the user presses a button on the GUI to provide redundant feedback.[36] The final auditory cue implemented in this iteration of SurfTacs is a jet turbine sound for each DDG in the scene. The turbine sound simulates changes in throttle by adjusting volume and pitch and thus provides auditory feedback to the execution of lee-helm orders.

D. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

As noted earlier in this chapter, the message window provides the only reference to dialog between the user and the helmsman/lee-helmsman. Future research should be applied to creating a verbal interaction between these actors by exploring of the areas of auditory cueing, voice synthesis and voice recognition. Auditory cueing may be provided in the form of previously recorded words and phrases which are fused together in various ways. Voice synthesis dynamically creates a verbal representation of an arbitrary string of tokens (phrases, words and letters). Finally, voice recognition takes verbal input from the user and converts it to a machine understandable format. Auditory cueing, voice synthesis and voice recognition offer potential advantages but a review of their technological limitations must be carefully considered and is thus recommended as a future thesis research topic in and of itself.

IX. SUMMARY

A. CONCLUSION

SurfTacs represents more to the Navy than a single graduate student's perseverance over a year long struggle. SurfTacs has a greater meaning than the end product of this or any future iteration. SurfTacs is an example to the Navy that training needs may be solved in-house by applying the benefits of open source media and game-based training. It is strongly encouraged for the Navy to devote manpower resources to the development of a small cadre of personnel to build upon the open source paradigm. The Naval Postgraduate School offers an excellent opportunity to train and screen future members of an in-house application development team.

The open source paradigm offers numerous advantages to the Navy's simulation needs. Open source is a scalable resource that maximizes the benefits of software reuse. Applications produced in this manner will be more cost-effective through the intrinsic value of free distribution. Open source is a tool the Navy must leverage in order to meet the growing demand for tactical surface simulation and to revolutionize Naval training in general. Whether there is continued interest in SurfTacs or not, the Navy should not miss this grand opportunity to innovate by exploiting the open source paradigm.

B. FUTURE RESEARCH AND APPLICATION TO SURFACE TACTICS (IN GENERAL)

Throughout this thesis areas of potential future research and application to surface tactics were identified. It is the author's hope that future surface warfare officers will also see the potential of using SurfTacs as a vehicle for their research. The future of navy surface warfare training is in your hands. Understand that you will fail more often than you succeed, but that is the only way you will truly learn.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Dorsey, J., "Warship Training Tactic Under Scrutiny," in The Virginia Pilot, [online magazine] (2 November 2005 [cited 20 Dec 2005]); available from the World Wide Web @ http://home.hamptonroads.com/stories/story.cfm?story=94602&ran=100160

2. Grassi, C., *A Task Analysis of Pier Side Ship-Handling for Virtual Environment Ship-Handling Simulator Scenario Development*, (Master's Thesis, Naval Postgraduate School, 2000), 5.

3. Norris, S., *A Task Analysis of Underway Replenishment for Virtual Environment Ship-Handling Simulator Scenario Development*, (Master's Thesis, Naval Postgraduate School, 1998), 7.

4. Grassi, C., *A Task Analysis of Pier Side Ship-Handling for Virtual Environment Ship-Handling Simulator Scenario Development*, (Master's Thesis, Naval Postgraduate School, 2000), 8.

5. . Norris, S., *A Task Analysis of Underway Replenishment for Virtual Environment Ship-Handling Simulator Scenario Development*, (Master's Thesis, Naval Postgraduate School, 1998), 16-17.

6. Grassi, C., *A Task Analysis of Pier Side Ship-Handling for Virtual Environment Ship-Handling Simulator Scenario Development*, (Master's Thesis, Naval Postgraduate School, 2000), 8-9.

7. Reid, D., "Open Source Turns Money Spinner," in BBC News [online magazine](6 November 2005 [cited 20 December 2005]); available from the World Wide Web @ http://news.bbc.co.uk/2/hi/programmes/click_online/4407742.stm.

8. Reid, D., "Open Source Turns Money Spinner," in BBC News [online magazine](6 November 2005 [cited 20 December 2005]); available from the World Wide Web @ http://news.bbc.co.uk/2/hi/programmes/click_online/4407742.stm.

9. Open Source Initiative [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.opensource.org.

10. Open Source Initiative [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.opensource.org.

11. Source Forge, "SourceForge.net: What is SourceForge.net?" [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://sourceforge.net/docs/about.

12. Source Forge, “SourceForge.net: What is SourceForge.net?” [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://sourceforge.net/docs/about.

13. Linux, “Introduction to Linux and Linux.com,” [electronic bulletin board](21 July 2004 [cited 20 December 2005]); available from the World Wide Web @ http://www.linux.com/article.pl?sid=02/03/09/1727250

14. Kirriemuir, J., “Video Gaming, Education and Digital Learning Technologies,” in D-Lib Magazine [online magazine](February 2002 [cited 20 December 2005]); available from the World Wide Web @ http://www.dlib.org/dlib/february02/kirriemuir/02kirriemuir.html

15. Amory, A., Naicker, K., Vincent, J. and Adams, C., “The use of Computer Games as an Educational Tool: 1. Identification of Appropriate Game Types and Game Elements,” in British Journal of Educational Technology [online journal] (30(4) 1999: 311-322 [cited 20 December 2005]); available from the World Wide Web @ http://72.14.203.104/search?q=cache:S23OWECP4C8J:www.nu.ac.za/biology/staff/amory/bjet30.rtf+amory+curiosity&hl=en&client=firefox-a

16. Rusbult, C., “Motivations for Learning and Strategies for Learning,” [electronic bulletin board](2002 [cited 20 December 2005]); available from the World Wide Web @ http://www.asa3.org/ASA/education/learn/motives.htm

17. “Waterfall Model,” in Wikipedia [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://en.wikipedia.org/wiki/Waterfall_model

18. “Spiral Model,” in Wikipedia [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://en.wikipedia.org/wiki/Spiral_model

19. “Iterative Design,” in Wikipedia [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://en.wikipedia.org/wiki/Iterative_design

20. “Extensibility,” in Merriam-Webster Online Dictionary [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.m-w.com/dictionary.htm

21. “Object-oriented Programming,” in Wikipedia [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://en.wikipedia.org/wiki/Object-oriented_programming

22. Delta3D [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://delta3d.org

23. CEGUI [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.cegui.org.uk

24. OpenAL [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.openal.org

25. OpenSceneGraph [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.openscenegraph.org

26. OpenGL [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.opengl.org

27. Bennett, S., Skelton, J. and Lunn, K., *Schaum's Outline Series: UML*, (New York: McGraw-Hill, 2001), 47-110.

28. "Osgconv" in Open Scene Graph [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.openscenegraph.org/index.php?page=UserGuides.Osgconv

29. Open Dynamics Engine [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://ode.org

30. "Bernoulli Principle," in Wikipedia [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://en.wikipedia.org/wiki/Bernoulli%27s_principle

31. Wickens, C., Lee, J., Liu, Y. and Gordon Becker, S., *An Introduction to Human Factors Engineering*, 2nd Ed. (Upper Saddle River: Pearson Prentice Hall, 2004), 121-154.

32. Wickens, C., Lee, J., Liu, Y. and Gordon Becker, S., *An Introduction to Human Factors Engineering*, 2nd Ed. (Upper Saddle River: Pearson Prentice Hall, 2004), 417

33. Wickens, C., Lee, J., Liu, Y. and Gordon Becker, S., *An Introduction to Human Factors Engineering*, 2nd Ed. (Upper Saddle River: Pearson Prentice Hall, 2004), 193

34. Wickens, C., Lee, J., Liu, Y. and Gordon Becker, S., *An Introduction to Human Factors Engineering*, 2nd Ed. (Upper Saddle River: Pearson Prentice Hall, 2004), 194

35. USS ROOSEVELT (DDG 80) Official Website [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.ddg-roosevelt.navy.mil/albums/precomm/underwaybinmage.jpg

36. Wickens, C., Lee, J., Liu, Y. and Gordon Becker, S., *An Introduction to Human Factors Engineering*, 2nd Ed. (Upper Saddle River: Pearson Prentice Hall, 2004), 127

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

Amory, A., Naicker, K., Vincent, J., and Adams, C. "The use of Computer Games as an Educational Tool: 1. Identification of Appropriate Game Types and Game Elements," in British Journal of Educational Technology [online journal]. 30(4) 1999: 311-322 [cited 20 December 2005]. Available from the World Wide Web @ http://72.14.203.104/search?q=cache:S23OWECP4C8J:www.nu.ac.za/biology/staff/ amory/bjet30.rtf+amory+curiosity&hl=en&client=firefox-a

Bennett, S., Skelton J., and Lunn, K. *Schaum's Outline Series: UML*. New York: McGraw-Hill, 2001.

"Bernoulli Principle," in Wikipedia [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://en.wikipedia.org/wiki/Bernoulli%27s_principle

CEGUI [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.cegui.org.uk

Delta3D [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://delta3d.org

Dorsey, Jack. "Warship Training Tactic Under Scrutiny," in The Virginia Pilot, [online magazine]. 2 November 2005 [cited 20 Dec 2005]. Available from the World Wide Web @ http://home.hamptonroads.com/stories/story.cfm?story=94602&ran=100160

"Extensibility," in Merriam-Webster Online Dictionary [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.m-w.com/dictionary.htm

Grassi, Charles. *A Task Analysis of Pier Side Ship-Handling for Virtual Environment Ship-Handling Simulator Scenario Development*. Master's Thesis, Naval Postgraduate School, 2000.

"Iterative Design," in Wikipedia [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://en.wikipedia.org/wiki/Iterative_design

Kirriemuir, John. "Video Gaming, Education and Digital Learning Technologies," in D-Lib Magazine [online magazine]. February 2002 [cited 20 December 2005]. Available from the World Wide Web http://www.dlib.org/dlib/february02/kirriemuir/ 02kirriemuir.html

Linux, “Introduction to Linux and Linux.com,” [electronic bulletin board]. 21 July 2004 [cited 20 December 2005]. Available from the World Wide Web @ http://www.linux.com/article.pl?sid=02/03/09/1727250

Norris, Steven. *A Task Analysis of Underway Replenishment for Virtual Environment Ship-Handling Simulator Scenario Development*. Master’s Thesis, Naval Postgraduate School, 1998.

“Object-oriented Programming,” in Wikipedia [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://en.wikipedia.org/wiki/Object-oriented_programming

OpenAL [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.openal.org

OpenGL [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.opengl.org

OpenSceneGraph [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.openscenegraph.org

Open Dynamics Engine [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://ode.org

Open Source Initiative [electronic bulletin board][cited 20 December 2005]; available from the World Wide Web @ http://www.opensource.org

“Osgconv” in Open Scene Graph [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.openscenegraph.org/index.php? page=UserGuides.Osgconv

Reid, David. “Open Source Turns Money Spinner,” in BBC News [online magazine]. 6 November 2005 [cited 20 December 2005]. Available from the World Wide Web @ http://news.bbc.co.uk/2/hi/programmes/click_online/4407742.stm

Rusbult, C. “Motivations for Learning and Strategies for Learning,” [electronic bulletin board]. 2002 [cited 20 December 2005]. Available from the World Wide Web @ http://www.asa3.org/ASA/education/learn/motives.htm

Source Forge, “SourceForge.net: What is SourceForge.net?” [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://sourceforge.net/docs/about

“Spiral Model,” in Wikipedia [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://en.wikipedia.org/wiki/Spiral_model

USS ROOSEVELT (DDG 80) Official Website [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://www.ddg-roosevelt.navy.mil/albums/precomm/underwaybinmage.jpg

"Waterfall Model," in Wikipedia [electronic bulletin board]. [cited 20 December 2005]. Available from the World Wide Web @ http://en.wikipedia.org/wiki/Waterfall_model

Wickens, C., Lee, J., Liu, Y., and Gordon Becker, S., *An Introduction to Human Factors Engineering*. 2nd Ed. Upper Saddle River: Pearson Prentice Hall, 2004.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Chairman, Code CS
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA

4. Rudolph P. Darken, Ph.D. Code CS/Dk
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA